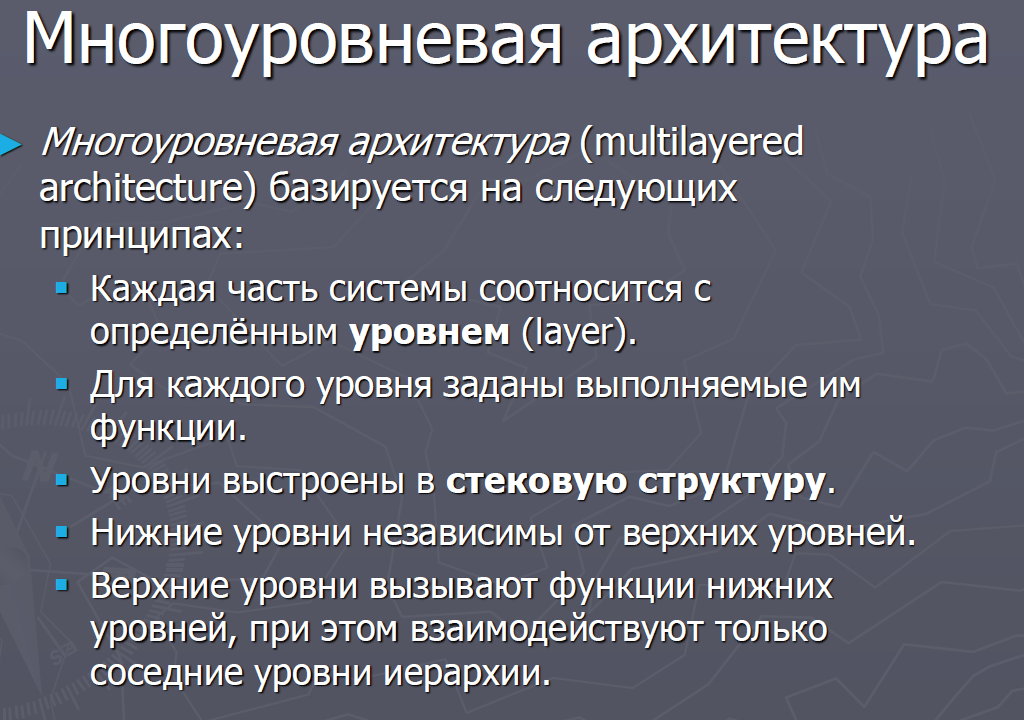
1. **Для чего используют многоуровневые архитектуры?**

Система делится на уровни, каждый из которых взаимодействует лишь с двумя соседними. Поэтому запросы к БД, которая обычно располагается в самом конце цепочки взаимодействия, проходят последовательно сквозь каждый «слой».

Архитектура [не подразумевает](https://www.oreilly.com/ideas/software-architecture-patterns/page/2/layered-architecture) какое-то обязательное количество уровней — их может быть три, четыре, пять и больше. Чаще всего используют трехзвенные системы: с уровнем представления (клиентом), уровнем логики и уровнем данных.

Каждый уровень этой архитектуры выполняет строго ограниченный набор функций (которые не повторяются от слоя к слою) и не знает о том, как устроены остальные уровни. Поэтому «содержимое» уровней можно изменять без риска глобальных конфликтов между слоями.



1. **Опишите назначение слоев многоуровневой архитектуре. Business layer (уровень бизнес-логики), Data Access layer (уровень доступа к данным).**

**Business layer** (уровень бизнес-логики): содержит набор компонентов, которые отвечают за обработку полученных от уровня представлений данных, реализует всю необходимую логику приложения, все вычисления, взаимодействует с базой данных и передает уровню представления результат обработки.

**Data Access layer** (уровень доступа к данным): хранит модели, описывающие используемые сущности, также здесь размещаются специфичные классы для работы с разными технологиями доступа к данным, например, класс контекста данных Entity Framework. Здесь также хранятся репозитории, через которые уровень бизнес-логики взаимодействует с базой данных.

1. **Поясните назначение и принцип использования паттерна Repository?**

«Репозиторий» — один из наиболее часто используемых паттернов проектирования при работе с базами данных (БД). Он позволяет отделить программную логику, работающую непосредственно с БД, от всей остальной программы выступая посредником меду ними с помощью интерфейса во многом схожего с коллекциями.

В чем преимущество использования паттерна «Репозиторий»?

Отделение программной логики, работающей непосредственно с БД от всей остальной программы. Если требуется изменить логику работы с БД или даже тип БД (например, перенос БД с MS SQL Server на MySQL), то внесение изменений осуществляется локально и централизованно. Нет необходимости вносить многочисленные правки по всей программе с вероятностью что-либо пропустить и тем самым спровоцировать ошибку в работе программы.

С помощью «Репозитория» можно значительно упростить и алгоритмы по работе с БД в остальной программе. Чаще всего для выполнения какой-либо операции с БД достаточно вызвать один из методов класса, реализующего этот паттерн.

1. **Поясните назначение и принцип использования паттерна Unit of Work?**

Паттерн Unit of Work позволяет упростить работу с различными репозиториями и дает уверенность, что все репозитории будут использовать один и тот же контекст данных.

1. Опишите основные особенности библиотеки Entity Framework?
2. **Какие преимущества обеспечивает использование слабосвязанного кода?**

Самое большое преимущество заключается в том, что введение изменений в один модуль не нарушает другие модули непредсказуемым образом.

Другим преимуществом является возможность замены компонентов более легко, когда у вас более одной реализации.